UNIDAD Nº 1: ARQUITECTURA DE COMPUTADORES

OBJETIVOS: Entender desde el punto de vista funcional, la organización de los componentes de computadoras, y las distintas arquitecturas que estas combinaciones definen.

1.1 Algunas definiciones:

Existen distintas definiciones para explicar que es una arquitectura de computador, entre ellas tenemos:

- "Arte de diseñar una máquina con la cual sea agradable trabajar "(Caxton Foster - 1970)
- "Determinar componentes, funciones de los componentes y reglas de interacción entre ellos" (N. Prassard - 1981)
- "La estructura de la computadoras que el programador necesita conocer con el objeto de escribir programas en lenguaje de máquina correctos" (Informe final del proyecto CFA, Arquitectura de Familias de Computadoras)

Cada definición da una versión desde un punto de vista que depende del interés del estudio, para nosotros, la arquitectura de una computadora la definiremos como, *la composición y combinación funcional de los distintos elementos de un computador.*

Esto es, desde el punto de vista de Sistemas de Información (Unidad I de sistemas de procesamiento de datos), las distintas aplicaciones (salidas), determinaban los requerimientos del bloque funcional del proceso y por supuesto los datos de entrada.

La definición del proceso, de esta forma, indica como deben interactuar las distintas variables, y para que esto suceda, las mismas deben estar soportadas en una estructura diseñada a tal efecto, esta estructura que tiene en cuenta los componentes de la computadora, la función individual de los mismos, y la función colectiva de acuerdo a los estímulos que recibe de las entradas, es lo que definimos como Arquitectura de las computadoras.

En definitiva una arquitectura esta dada por:

- Componentes
- Interconexión de sus componentes
- Interacción entre sus componentes.
- Diseño de los componentes

- Forma de usar todos estos aspectos en beneficio de una tarea o sea su operación.

1.2 Arquitecturas secuenciales

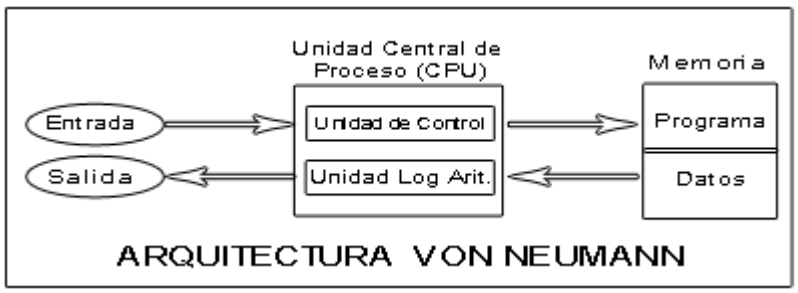
Hasta ahora las computadoras vistas responden a una estructura general, que es la determinada por Von Neumann (1945), cuyos conceptos rigen casi todos los diseños de las computadoras actuales, y son:

- Programa almacenado y datos almacenados (esto significa que tanto programas como datos deben residir en una memoria, común o no, pero que deben estar representados internamente para poder ser ejecutados y tratados)
- Flujo de cómputos secuencial
- Representación binaria de la información (o sea, que datos y programas se representan de igual forma)
- Flujo de información interno en paralelo, en vez de en serie (esto claramente indica que la cantidad de circuitos debe aumentar hasta tantos bits en paralelo como se pretendan que se muevan juntos en un flujo de datos)
- No paralelismo de las operaciones

Von Neumann describió como una computadora comprende cinco unidades, una entrada, una salida, un procesador aritmético, una unidad de control y una memoria, y de que manera, siguiendo el criterio de la unidad de control, las instrucciones se pueden almacenar en la misma memoria interna que los datos y pueden ser interpretados por esa unidad de control de la misma forma que los datos por la unidad aritmética.

1.2.1 Proceso de Cómputo secuencial

El concepto más difundido cuando se habla de una arquitectura Von Neumann es el del proceso de cómputo.



En esta arquitectura el programa se almacena en memoria como una secuencia de instrucciones y se ejecuta extrayendo las mismas en una secuencia e interpretándolas. Por lo tanto el curso

que sigue el cómputo viene dado por la secuencia de las instrucciones del programa (es decir, el flujo de control del programa).

Para lograr esta arquitectura secuencial, es necesario disponer a nivel de hardware de un registro que tenga la dirección de la próxima instrucción, este se denomina *contador de programa*, y es un elemento que se encuentra dentro de la unidad de control.

Este elemento es el que indica la dirección de la próxima instrucción a extraer de la memoria, cargar en la unidad de control y ejecutar.

En este punto se observa claramente lo que se definió como arquitectura, un conjunto de componentes cuya función individual y operación colectiva realizaban una cierta aplicación, en este caso secuencial.

1.3 Estructuras para una arquitectura secuencial y de monoprocesador

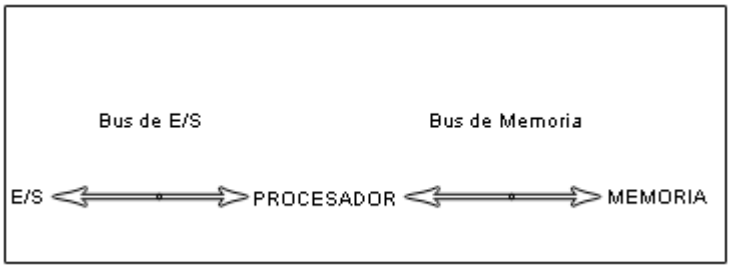
Se define como estructura:

La interconexión entre los distintos componentes. Por ejemplo que el diseño tenga una CPU, una memoria y un canal de E/S y que la CPU se conecte entre la memoria y el canal de E/S.

Como ejemplo de distintas estructuras para una misma Arquitectura Secuencial y de monoprocesador tenemos de dos buses, de dos buses con canales de E/S y de un bus.

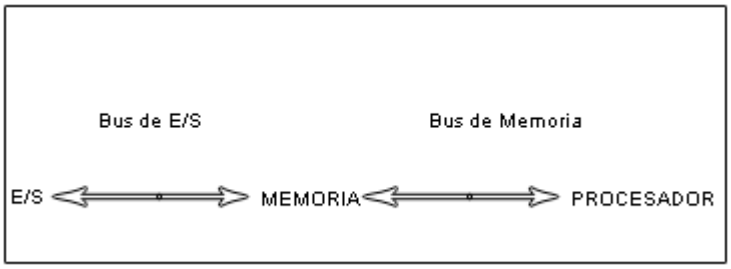
1.3.1 Dos Buses

Todo dato de E/S pasará por el procesador, luego las operaciones y transferencias de E/S se realizan bajo control directo del procesador.



1.3.2 Dos Buses (Alternativa) (Canales)

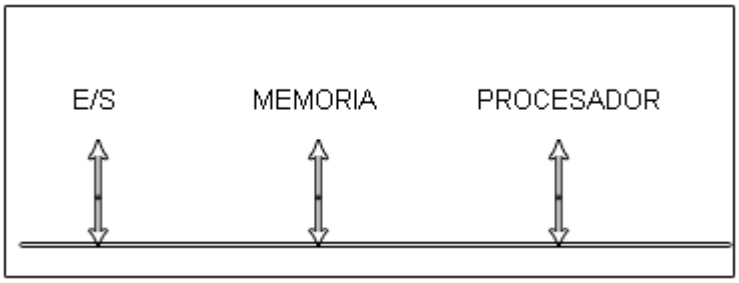
Aquí las operaciones de E/S y transferencia se realizan Bajo el control de procesadores de E/S (canales de E/S)



Generalmente el procesador pasa a los canales los parámetros y el control de las operaciones de E/S.

1.3.3 Un Bus

Aquí el bus se puede utilizar para una sola transferencia por vez (obvio hay un solo bus) o sea que sólo dos unidades funcionales pueden estar haciendo uso de él.



En esta estructura es fácil incorporar nuevos periféricos. Es una estructura de bajo costo y es muy utilizada en microcomputadoras.

A pesar de estas dificultades de estructura que hacen al rendimiento, los principios fundamentales son independientes de ellas y siguen siendo máquinas secuenciales.

1.4 Arquitecturas para el procesamiento Paralelas

Las arquitecturas paralelas son aquellas que están soportadas por el Procesamiento Paralelo, el cual se basa en la explotación de sucesos concurrentes en un proceso de cómputos.

La concurrencia implica paralelismo, simultaneidad y pipelining.

Sucesos Paralelos: ocurren en múltiples recursos durante el mismo intervalo.

Sucesos simultáneos: ocurren en el mismo instante.

Sucesos Pipeline: ocurren en lapsos superpuestos.

Como se ha dicho, las arquitecturas paralelas son las que permiten la explotación de los sucesos concurrentes, los cuales se pueden dar a distintos niveles, esto es, a nivel de programa, de instrucción o de dato.

Podemos tener muchos programas ejecutándose concurrentemente, lo que significa Multiprogramación, podemos tener un programa, cuyas instrucciones se ejecuten concurrentemente, y también instrucciones cuyas operaciones de ejecución se ejecuten concurrentemente.

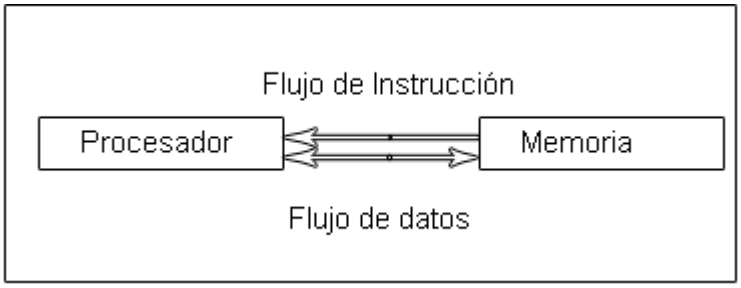
A nivel de Hardware el paralelismo puede obtenerse de distintas maneras:

- Varias computadoras conectadas para realizar una tarea común.
- Varios procesadores dentro de una computadora.
- Varias computadoras conectadas a un canal para ofrecer recursos.
- Procesadores capaces de trabajar en forma Pipeline.

1.4.1 Tipos de procesamiento en paralelo.

A partir del estudio de las etapas de una instrucción TI – DI – TO – EI (traer instrucción, decodificarla, traer operandos y ejecutar la instrucción), y su posterior tratamiento aparecen las distintas arquitecturas.

Estas operaciones definen un flujo de instrucciones que se desplaza desde la memoria al procesador y un flujo de datos que se desplaza entre la memoria y el procesador, como en la figura.



Combinando el flujo de instrucciones y el flujo de datos, surgen los cuatro grupos de procesadores posibles, los que manejan una sola instrucción y un solo dato, los de una sola instrucción pero múltiples datos, los de múltiples instrucciones y un solo dato y los de múltiples instrucciones y datos.

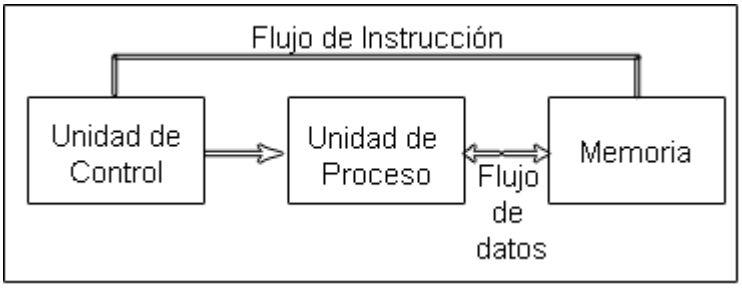
La arquitectura más básica es la que procesa un solo dato y una sola instrucción a la vez, es decir sigue la secuencia de las cuatro etapas.

1.4.2 Arquitecturas SISD – Una sola instrucción y un solo dato a la vez.

Normalmente se las conoce como SISD (Single Instruction Single Data).
Por ejemplo un programa con las siguientes características:

```
For (I=1; I<10; I++)
      A[I]= A[I]*3
End For
```

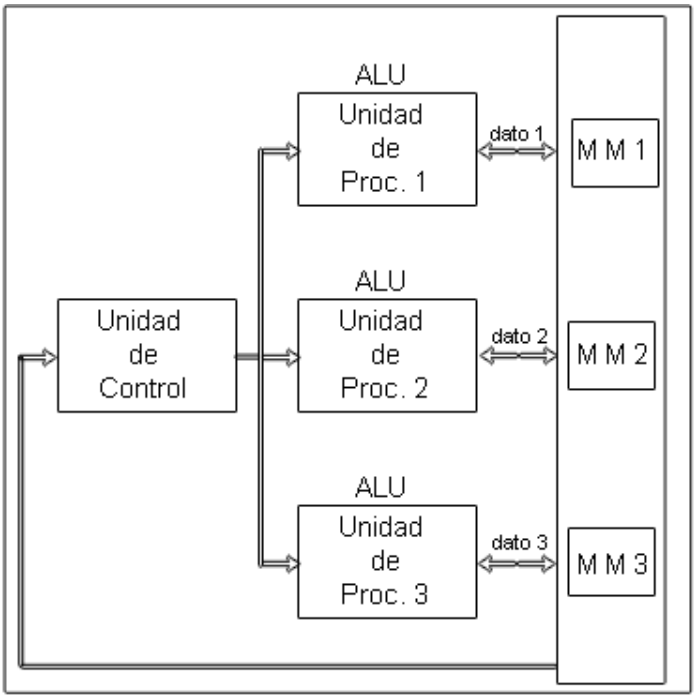
Las diez veces que se pasa por la asignación, serán 10 instrucciones completamente independientes una de otra.



1.4.3 Arquitecturas SIMD – Una sola instrucción y múltiples datos a la vez.

Si tomamos el ejemplo anterior y colocamos diez unidades de procesamiento simultáneas bajo el control de una única unidad de control, ejecutaríamos las diez instrucciones en forma simultánea, con lo cual se ahorra mucho tiempo.

Como se observa en la figura, esta arquitectura, tiene la siguiente estructura, una memoria, una unidad de control y varias unidades de ejecución que depositarán sus datos en la memoria.



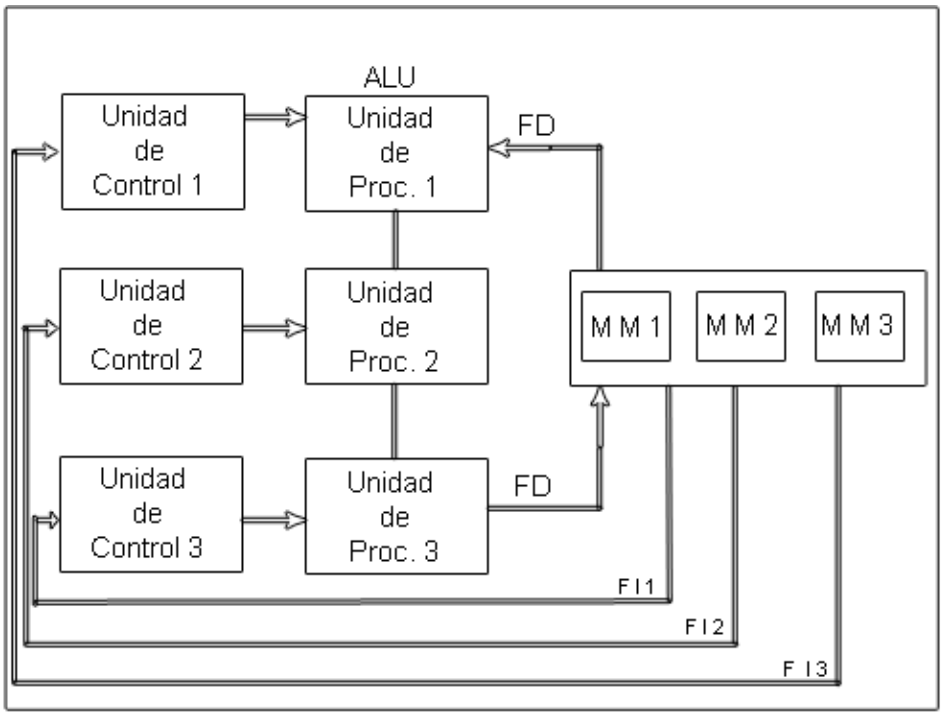
Estas unidades de ejecución son las ALU, por ejemplo el microprocesador P5 ya soporta este tipo de arquitecturas en paralelo, porque viene provisto de 2 ALU.

A estas arquitecturas se las denomina SIMD (single Instruction Multiple Data), o sea una sola instrucción que ataca a muchos datos.

1.4.4 Arquitectura MISD – Muchas Instrucciones y un único flujo de datos a la vez.

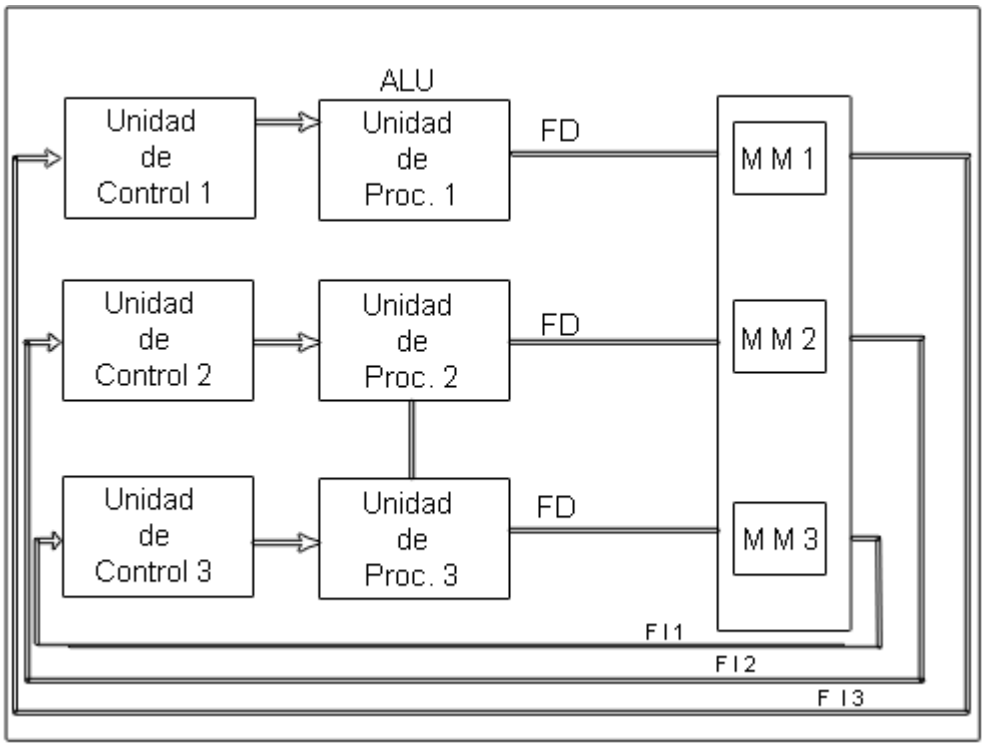
Como se ve en la figura, en este caso tenemos varias unidades de control, con varias unidades de ejecución, existen distintos flujos de instrucciones y un único flujo de datos. O sea tienen múltiples instrucciones con un solo dato.

Este es un esquema teórico nada más.



1.4.5 Arquitecturas MIMD – Muchas instrucciones y múltiples datos a la vez.

En este caso, como se ve en la siguiente figura, tenemos varias unidades de control con varias unidades de ejecución. Es decir se ejecutan muchas instrucciones con muchos datos siendo este un caso típico de multiprocesamiento.



MIMD (Multiple Instruction Multiple Data)

1.5 Concepto de Acople

Existen similitudes entre los sistemas multiprocesadores y multicomputadoras debido a que ambos fueron pensados con un mismo objetivo: dar soporte a operaciones concurrentes en el sistema. Sin embargo, existen diferencias importantes basadas en el alcance de los recursos compartidos y la cooperación en la solución de un problema.

Un sistema multicomputador consiste de diversas computadoras autónomas que pueden o no comunicarse entre sí.

Un sistema multiprocesador está controlado por un sistema operativo que provee la interacción entre los procesadores y sus programas a nivel de dato, proceso y archivo.

Existen dos modelos arquitectónicos diferentes para los sistemas multiprocesadores: fuertemente acoplados y débilmente acoplados. Los sistemas fuertemente acoplados se comunican a través de una memoria común.

Puede existir una pequeña memoria local o un buffer de alta velocidad (caché) en cada procesador.

Existe una completa conectividad entre los procesadores y la memoria. Esta conectividad puede alcanzarse insertando una red de interconexión entre los procesadores y la memoria o mediante una memoria multipuertas.

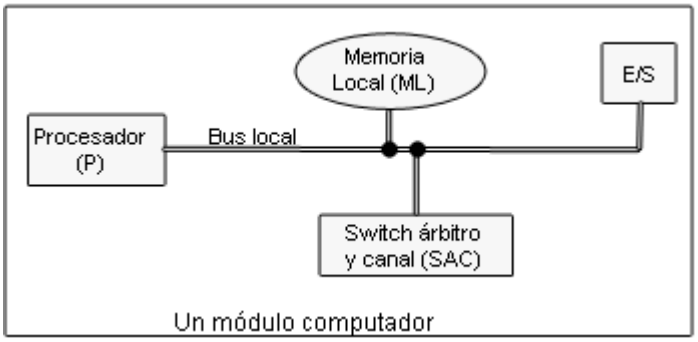
Uno de los factores que limitan el crecimiento de los sistemas fuertemente acoplados es la degradación debido a la contención de memoria que ocurre cuando dos o más procesadores intentan acceder a la misma unidad de memoria concurrentemente.

Puede reducirse el grado de conflictividad incrementando el grado de interleaving. Sin embargo, esto debe acompañarse de una cuidadosa asignación de los datos a los módulos de memoria.

Los sistemas multiprocesadores débilmente acoplados no tienen, en general, el grado de conflicto sobre la memoria de los fuertemente acoplados.

1.5.1 Acoplamiento débil y acoplamiento fuerte.

En este sistema cada procesador tiene un conjunto de dispositivos de E/S y una gran memoria local a donde accede para obtener la mayoría de sus datos e instrucciones.



Nos referimos al procesador, sus dispositivos de E/S y su memoria como el módulo computador.

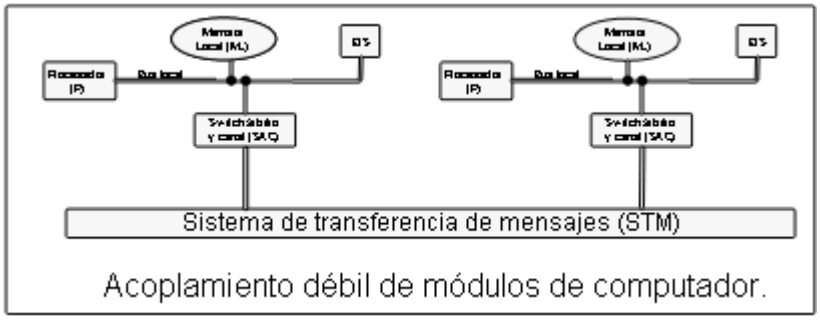
Los procesos que se ejecutan en diferentes procesadores se comunican intercambiando mensajes a través de un sistema de transferencia de mensajes. El grado de acoplamiento en tales sistemas es realmente muy débil, de allí que se los conozca también como sistemas distribuidos.

El factor determinante del grado de acoplamiento es la topología de la comunicación del sistema de transferencia de mensajes asociado.

Los sistemas débilmente acoplados son eficientes cuando la interacción entre las tareas en mínima.

Los sistemas fuertemente acoplados pueden soportar una gran interacción entre las tareas sin un deterioro significativo de la performance.

La figura muestra un ejemplo de un módulo computador de un multiprocesador débilmente acoplado no jerárquico.



Acoplamiento débil de módulos de computador.

Consiste en un procesador, una memoria local, dispositivos de E/S locales y una interfase a otros módulos computadores.

La interfase puede contener un switch árbitro y un canal. La primera figura muestra también la conexión entre los módulos computador y el sistema de trasferencia de mensajes (STM).

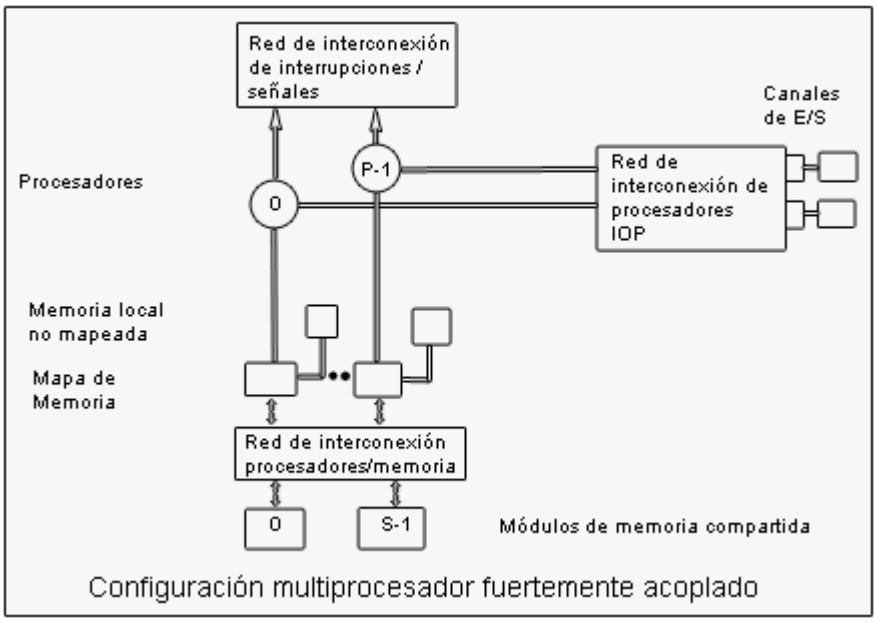
Si los pedidos para los dos o más módulos computador colisionan al acceder un segmento físico del STM, el árbitro es el responsable de elegir uno de los pedidos simultáneos de acuerdo a una determinada disciplina de servicio.

Es también responsable de hacer esperar los otros pedidos hasta que se complete la atención del pedido actual.

El canal que se encuentra dentro del SAC puede tener una memoria de comunicación de alta velocidad a efectos de bufferizar los bloques de trasferencia de mensajes. La memoria de comunicación es accesible para todos los procesadores, en la próxima figura podemos ver un esquema de multiprocesador fuertemente acoplado.

Esta consiste en P procesadores, S módulos de memoria y D canales de entrada/salida. Estas unidades están conectadas mediante tres redes de interconexión, a saber, la red de interconexión entre los procesadores y los IOP, y la red de interconexión de interrupciones-señales.

Los conflictos de acceso a memoria por varios procesadores son resueltos por la red de interconexión procesador-memoria. Para evitar excesivos conflictos, la cantidad de módulos de memoria S es generalmente tan grande como P.



Configuración multiprocesador fuertemente acoplado

Otro método para deducir el grado de conflicto es asociar un área de almacenamiento reservada para cada procesador. Esta es la memoria local no mapeada que se usa para almacenar código Kernel y tablas del sistema operativo muy utilizadas por los procesos que se ejecutan en el procesador.

Se puede agregar también a esta configuración una memoria caché propia de cada procesador a fin de disminuir las referencias a memoria principal.

La red de interconexión de interrupciones-señales permite que cada procesador envíe directamente una interrupción a otro procesador. La sincronización entre procesos se ve facilitada por esta red. Esta red puede actuar como un procesador de fallas ya que puede enviar una alarma originada por hardware a los procesadores que si funcionan.

El conjunto de procesadores puede ser homogéneo o heterogéneo. Es homogéneo si los procesadores son funcionalmente idénticos.

Pero aún siendo homogéneos pueden ser simétricos o asimétricos dependiendo de que dos unidades funcionalmente idénticas difieran en cuanto a dimensiones tales como la accesibilidad de E/S, performance o confiabilidad. Esta última configuración de la figura, como procesador fuertemente acoplado es también conocida como sistema de multiprocesadores diádicos.

1.6 Paralelismo con una sola CPU

Salvo en el caso de la SISD, las demás arquitecturas permiten el paralelismo por la simple utilización adecuada de la estructura del hardware, sin embargo cuando disponemos de una única CPU, el desarrollo de un procesamiento en paralelo se debe a otros tipo de técnicas de aplicación.

Procesamiento en paralelo es un término utilizado para denotar operaciones simultáneas en la CPU con el fin de aumentar su velocidad de cómputo. El lugar de procesar secuencialmente como en las arquitecturas convencionales, un procesador paralelo, realiza tareas de procesamiento de datos e instrucciones concurrentemente.

Para lograr concurrencia en un sistema con un solo procesador, se utilizan técnicas de paralelismo que se consiguen multiplicando los componentes de hardware o técnicas de Pipeline.

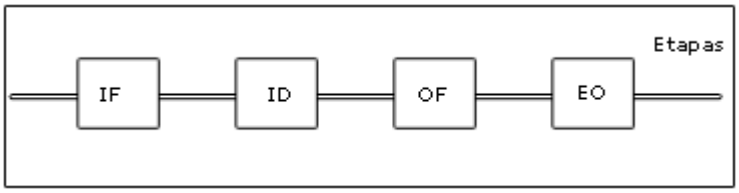
Un conmutador Pipeline hace operaciones superpuestas para explotar el paralelismo temporal. Un array procesador usa ALUs múltiples sincronizadas, para lograr paralelismo espacial. Un sistema multiprocesador logra paralelismos asincrónicos, a través de un conjunto de procesadores que interactúan y comparten recursos (periféricos, memorias, bases de datos, etc.).

El Pipeline es una forma económica de hacer paralelismo temporal en computadoras. La idea es la misma que la de las líneas de montaje de las plantas industriales. Se divide la tarea en una secuencia de subtareas, cada una de las cuales se ejecuta en una etapa de hardware especializada que trabaja concurrentemente con otras de las etapas del Pipeline. Esto permite aumentar el Throughput del sistema de forma considerable.

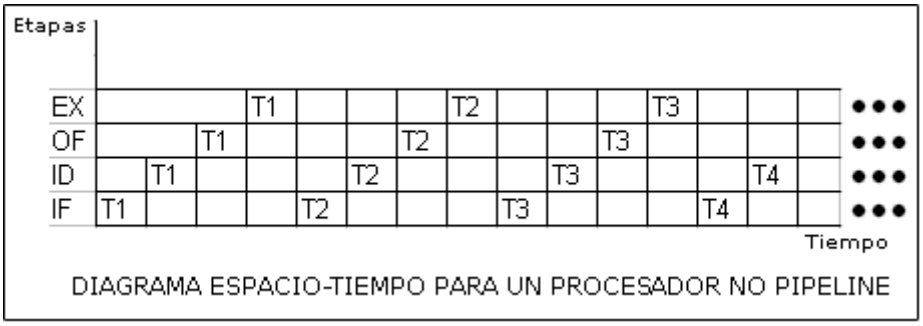
1.6.1 Computadores Pipeline

Veamos el ejemplo de un Pipeline de 4 etapas: El proceso de ejecución de una instrucción en un computador digital envuelve 4 pasos principales: Levantar la instrucción de memoria (Instruction Fech - IF); Identificar las operaciones que deben efectuarse (Instruction Decoding - ID); Levantar los operadores si son necesarios en la ejecución (Operand Feach – OF); y ejecutar la operación aritmética-lógica que ha sido decodificada.

Antes de comenzar a ejecutar una nueva instrucción debe completarse estos 4 pasos.



La idea de un computador Pipeline, como vemos en la figura: hay 4 etapas IF, ID, OF Y EX, ordenadas de forma de una cascada lineal. Las instrucciones sucesivas se ejecutan de la forma superpuesta. La diferencia entre la ejecución superpuesta de instrucciones y la ejecución no superpuesta secuencial se muestra en los diagramas de Espacio-Tiempo, en las siguientes figuras.



Cada columna del gráfico representa un ciclo de Pipeline, que es aproxima damente igual al tiempo que tarda la etapa mas lenta. A un computador sin Pipeline le toma cuatro ciclos de Pipeline completar una instrucción, en cambio un Pipeline produce un resultado de salida por cada ciclo luego de cargado el Pipe. El ciclo de instrucción ha sido reducido efectivamente a un cuarto del tiempo de ejecución original, por medio de las ejecuciones superpuestas.

Lo que hemos descripto anteriormente es un Pipeline de instrucción.

La máxima velocidad a la cual pueden ingresar las instrucciones al Pipeline depende exclusivamente del tiempo máximo requerido para atravesar una etapa y del número de ellas.

Hay ciertas dificultades que impedirán al Pipeline operar con su máxima velocidad. Los

segmentos pueden tomar tiempos diferentes para cumplir su función sobre los datos que llegan. Algunos segmentos son saltados por ciertas instrucciones. Por ejemplo una instrucción modo registro no necesita un cálculo de dirección; por otra parte dos o más segmentos pueden requerir acceso a memoria al mismo tiempo, haciendo que un segmento tenga que esperar hasta que el otro termine. (Los conflictos de acceso a memoria se resuelven a menudo utilizando técnicas de interleaving).

1.7 Arquitecturas DataFlow

Uno de los objetivos primordiales en el desarrollo de los sistemas es alcanzar las más altas velocidades y performance posibles. Este objetivo ha sido y sigue siendo alcanzado por dos medios:

- Explotando las necesidades tecnológicas de los componentes del computador.
- Adecuando estructuras y organizaciones del computador.

El intento de producir computadoras de arquitectura paralela con alta velocidad y performance obedece a las necesidades de resolver grandes problemas (la mayoría de características paralelas) tales como reconocimiento de patrones (pattern recognition), procesamiento de señales e imágenes (signal & image processing), problemas de inteligencia artificial, física nuclear, predicción, meteorología, control de tráfico aéreo, procesos de control de producción, etc.; muchos de los cuales deben resolverse en tiempo real.

Su eficiencia para resolver problemas en donde la estructura de datos no es tan regular, disminuye rápidamente, aún cuando son naturalmente paralelas. Estas computadoras son absolutamente ineficientes para procesar problemas seriales.

Otra desventaja sustancial es lograr la dependencia de la performance de estas computadoras del método de programación del problema. El programador debe conocer la estructura interna del computador para ser capaz de utilizar su paralelismo, y debe él mismo identificar el paralelismo en el problema o utilizar un programa especial para hacer esto último.

Muchos autores han llegado a la conclusión de que el problema central de la utilización del paralelismo reside en el modelo básico para expresar y escribir lo mismo.

El modelo de cómputo de Von Neumann aparece inapropiado a este respecto debido a que es un modelo serial.

Sorprendentemente todos los lenguajes de programación convencionales y las arquitecturas de las computadoras existentes, tanto SISD, SIMD o MIMD están basadas en este modelo de cómputo.

La naturaleza serial del modelo de Von Neumann y los problemas resultantes de ello son el principal obstáculo en la utilización del paralelismo, en las computadoras paralelas que procesan programas escritos en lenguajes convencionales de alto nivel.

Luego, se hace necesario crear primero un modelo de sistema computador el cual permita expresar algoritmos de un paralelismo natural.

Uno de tales modelos es el modelo DataFlow conocido como sistema DataDriver. Estrechamente relacionado con esto está el trabajo sobre utilización del principio de asignación única para expresar el paralelismo.

1.7.1 El modelo de cómputo DataFlow

Las diferencias entre el modelo cómputos DataFlow (DF) y el modelo convencional de cómputo Von Neumann (control Flow, CF), se observa fundamentalmente en que considera cada modelo como lo decisivo en el proceso de cómputo:

- En el modelo CF, es la secuencia de instrucciones.
- El DF, es la disponibilidad de los datos.

En una computadora convencional CF, el programa se almacena en la memoria como una secuencia de instrucciones. El programa se ejecuta extrayendo las sucesivas instrucciones de la memoria y ejecutándolas en el procesador.

Luego, el curso que sigue el cómputo está dado por la secuencia de instrucciones del programa (es decir, el flujo de control del programa).

No es posible ejecutar cualquier instrucción hasta que todas las instrucciones previas hayan sido ejecutadas. Si los operandos necesarios están disponibles, pueden existir en el programa instrucciones que podrían ejecutarse ya, pero ellas deben esperar a que les toque el turno en la secuencia.

Este es el obstáculo principal en la utilización de algoritmos de un paralelismo natural.

En las computadoras DF, el curso del cómputo está controlado por el flujo de datos en el programa. Una instrucción puede ejecutarse sólo cuando están disponibles todos sus operandos.

Estos operandos pueden ser obtenidos como datos de entrada o por ser resultados de instrucciones previas en el programa.

La precedencia de una instrucción respecto de otra está dada exclusivamente por la estructura natural del algoritmo que se está realizando y no depende de la ubicación de las instrucciones en la memoria.

Utilizando este modelo de cómputo es posible llegar a ejecutar tantas instrucciones en paralelo como puede el computador en cuestión.

Luego de la ejecución de la instrucción el resultado se distribuye a todas las instrucciones subsiguientes que han uso de ese resultado parcial del operando.

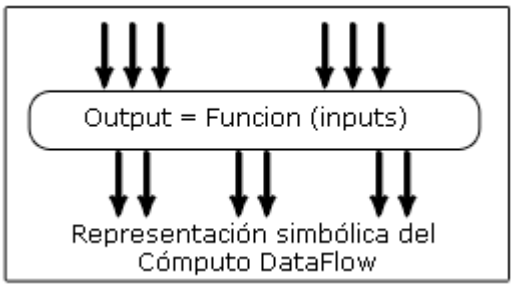
1.7.2 Representación de programas DataFlow

El diseño de modelo de cómputo DF, se explica mejor con un ejemplo.

Una de las mejores representaciones de un programa DataFlow es mediante un grafo dirigido el cual, puede ser visto como una simplificación del lenguaje DF.

Los nodos en el grafo representan operadores (instrucciones de programas) y los arcos representan caminos de datos unidireccionales a través de los cuales los resultados se transmiten en el programa.

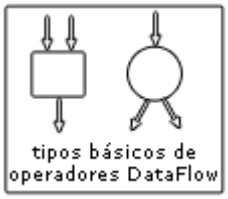
En general es posible representar la ejecución de una instrucción como una función sobre n inputs con m outputs, como se ve en la figura:



El operador consume sus inputs y libera un conjunto de resultados sobre los caminos outputs.

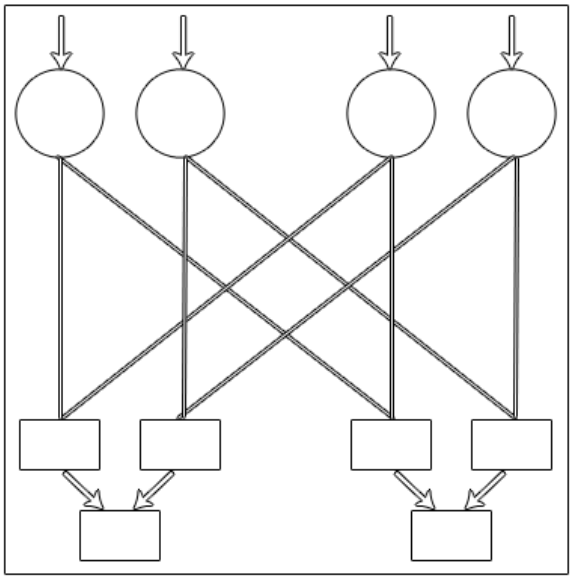
1.7.3 Tipos de operadores

Muchos de los lenguajes de DataFlow están limitados a dos tipos fundamentales de instrucciones (combinativa y separativa) con una cantidad de inputs y outputs iguales a los que se ve en la figura:



Las instrucciones combinativas tienen dos inputs y un output. El valor del output es una función de los valores de los inputs. Este operador puede ejecutar funciones sencillas como la suma, OR, AND, etc. o funciones complejas tales como la multiplicación, división o incluso procedimientos completos.

Las instrucciones separativas producen dos copias del dato de input que puede ser de tipo diferente.



Los caminos de los datos representan el flujo de los datos desde una instrucción hacia la próxima. Este es el sistema de comunicación de la ingeniería de implementación. Este sistema de comunicación puede ser un simple bus del sistema, pero puede también almacenar datos en memoria utilizando un sistema FIFO.